DAA/LANGLEY

Semi-Annual Report
Grant No. NAG-1-242

RESEARCH IN COMPUTER SCIENCE

Submitted to:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

Attention: Dr. John N. Shoosmith
ACD, MS 125

Submitted by:

J. M. Ortega
Professor and Chairman

Report No. UVA/528209/AM85/107
June 1985

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF APPLIED MATHEMATICS

UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA 22901

Semi-Annual Report

Grant No. NAG-1-242

RESEARCH IN COMPUTER SCIENCE

Submitted to:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

Attention: Dr. John N. Shoosmith
ACD, MS 125

Submitted by:

J. M. Ortega
Professor and Chairman

Department of Applied Mathematics
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/528209/AM85/107

June 1985

Copy No.____

This report summarizes work under NASA Grant NAG-1-242 for the period December 1, 1984 to June 1, 1985. During this period, eight graduate students were supported. The students their major area of interest, Langley contact, University of Virginia faculty advisor, and total period of support are summarized below.

Student	Area	Langley Contact	Advisor	Period
P. Ammann	Software Engineering	E. Senn	J. Knight	1/15/84-6/1/85
D. Bahler	Data Management	R. Fulton	J. Pfaltz	6/1/82-1/15/85
S. Brilliant	Software Engineering	E. Senn	J. Knight	6/1/83-
N. Fitzgerald	Concurrent Processing	O. Storassli	T. Pratt	1/15/84-1/15/85
C. Luan	Computer Graphics	D. Lansing	W. Martin	5/21/84-1/15/85
J. Marco	Software Engineering	E. Semn	W. Martin	5/28/84-
E. Poole	Concurrent Processing	J. Lambiotte	J. Ortega	6/1/83-
C. Vaughan	Concurrent Processing	J. Stroud	J. Ortega	1/15/85-

During this reporting period, sever l students who were currently or previously supported on this grant completed their m_sters degrees. These are listed below with the title of their thesis or project.

P. Ammann - An Experimental Evaluation of Simple Methods for Seeding Program Errors

S. Brilliant - Analysis of Faults in a Multi-Version Software Experiment

N. Fitzgerald - Implementation of a Parallel Programming Environment

V. Grine - Symbolic Execution of Concurrent Programs

P. Hurst - Executive Control Systems in the Engineering Design Environment

C. Luan - Two Computer Graphics Systems for Visualization of Pressure Distribution and Convective Density Particles

B. LoBracco -Distributed Wire Routing

E. Poole - An M-Step Incomplete Cholesky Preconditioned Conjugate Gradient Method for the CDC Cyber 203/205 Vector Computer

B. Smith - Extensions of the Domain Testing Method

L. St. Jean - Testing Version Independence in Multi-Version Programming

J. Taylor - Performance Analyzer of the PISCES System

Plans for the summer of 1985 call for six students to receive support. Four of these students will spend the summer at Langley Research Center while two others, Poole and Brilliant, will be at the University of Virginia. Both of these latter two students previously spent two summers at Langley. The summer program is summarized below.

Student	Area	Langley Contact	Advisor	Period
S. Brilliant	Software Engineering	E. Senn	J. Knight	6/1/83-
A. Cleary	Concurrent Processing	J. Stroud	J. Ortega	6/1/85-
S. Linde	Concurrent Processing	D. Dwoyer	J. Ortega	6/1/85-
J. Marco	Artificial Intelligence	N. Orlando	W. Martin	5/28/84-
E. Poole	Concurrent Processing	J. Lambiotte	J. Ortega	6/1/83-
C. Vaughn	Concurrent Processing	J. Stroud	J. Ortega	1/15/85-

We next give short summaries of the work performed during the reporting period.

Error Seeding as a Testing Method

Paul Ammann, Ph.D. Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

An experiment has been done in which simple syntactic alterations were introduced into program text. The experiment was carried out in order to evaluate the testing strategy known as error seeding. The experiment's goal was to determine if randomly placed syntactic manipulations can produce failure characteristics similar to those of the indigenous errors found within unseeded programs. A number of programs were available, each of which was written to the same specifications; thus the programs were intended to be functionally equivalent. The use of functionally equivalent programs allowed the influence of individual programmer styles to be removed as a variable from the error seeding experiment. Each of six different syntactic manipulations were introduced into each program and the mean times to failure for the seeded errors were observed in repeated trials over a fixed range of inputs. The seeded errors were found to have a broad spectrum of mean times to failure independent of the syntactic alteration used.

The experiment demonstrates that the use of simple syntactic techniques yields seeded errors that are arbitrarily difficulty to locate. In addition, several unexpected results indicate that some of the issues involved in error seeding have not been addressed previously. Specifically, certain of the seeded errors were benign; they had no effect on the program's functionality. Other seeded errors actually corrected indigenous errors. In addition, events inconsistent with assumptions underlying the test method were observed. Such results have clear implications for error seeding as a testing strategy.

A preliminary writeup of the experiment will appear in the Proceedings of the Eighth International Conference On Software Engineering to be held in London in August, 1985. A comprehensive writeup has been submitted to the IEEE Transactions on Software Engineering.

Knowledge Representation for Engineering Design

Dennis R. Bahler, Ph.D. Candidate in Computer Science
John L. Pfaltz, Professor of Computer Science

Mr. Bahler's new representation scheme for the modeling of three-dimensional objects, known as the B-spline cylinder, was refined, and the implementation was enhanced to permit the automatic construction of cylinder faces passing through any arbitrary set of interpolation points. These entities are comprised of interpolations among sets of cubic B-spline curves, and are similar to the generalized cylinders sometimes used in vision processing. Entities defined in this way have proven to be extremely efficient in both storage and computational cost, while being suitable for a variety of applications both in image understanding and geometric modeling. All the common primitives of a constructive solid geometry (CSG) modeler can be represented by this single entity class. In addition, more general free-form objects are also representable by exactly the same scheme and can function as user-defined primitives. Work has begun on the algorithms necessary to enable grouping of entities into more complex compound objects.

A number of promising new avenues of research were explored. Work was begun on defining a set of spatial and structural predicates over the domain of cylinder-objects, and a resolution-based theorem prover has been implemented for the purpose of employing automated deduction techniques in this domain. This system is now debugged. Also under investigation is the suitability of using a set of Euler operators together with a specified set of constraints that can then be automatically satisfied at each stage of construction of a primitive or complex object. Preliminary investigation has begun of the design of a deductive database containing geometric and topological information as well as non-geometric data pertaining to the object world.

Analysis of Faults in a Multi-Version Software Experiment

Susan S. Brilliant, Ph.D. Candidate in Computer Science
John C. Knight, Associate Professor of Computer Science

Multi-version programming has been proposed as a method of incorporating fault tolerance into software. Multiple versions of a program are prepared independently and executed in parallel. Their outputs are collected and examined by a voter, and, if they are not identical, it is assumed that the majority is correct.

The reliability improvement expected from the application of this method relies on the assumption that programs that have been developed independently will fail independently. An experiment was performed in which this fundamental axiom was tested. A total of twenty-seven versions of a program were prepared independently from the same specification at two universities and then subjected to one million tests. The results of the tests revealed that the programs were individually extremely reliable but that the number of tests in which more than one program failed was substantially more than expected. A detailed analysis of the faults in the versions was undertaken to determine the types of errors that result in correlated failures.

The conclusion drawn from this research is that multi-version programming must be used with care. Analysis of its reliability must include the effect of dependent errors. The use of programmers of diverse backgrounds and experience working in diverse environments cannot be relied upon to prevent such errors.

Implementation of a Parallel Programming Environment

Nancy J. Fitzgerald, Masters Candidate in Computer Science
Terrence W. Pratt, Professor of Computer Science

PISCES (Parallel Implementation of a Scientific Computing Environments) is a computer system intended for the solution of large scale problems in scientific and engineering computation. It is based on the use of MIMD parallel computation to achieve high computation rates. The system includes a programming environment, programming language, operating system, and machine architecture. Because the software provides an abstract "virtual machine" to the user, the precise details of the hardware and lower levels of the operating system software are not of concern to the user.

The base sequential language used in PISCES is FORTRAN. Language constructs have been added which allow the user to initiate tasks, and to communicate with other tasks via "message passing". A prototype system has been developed using UNIX as the underlying operating system. This implementation, which runs on a VAX under UNIX 4.1bsd, simulates the task and cluster level parallelism of the PISCES design on a uniprocessor. By using the UNIX "process" mechanism the tasks appear to be running in parallel, although, no actual parallel execution is possible.

Two Computer Graphics Systems for Visualization of Pressure Distribution and Convective Density Particles

Carol T. J. Luan, Masters Candidate in Computer Science
Worthy H. Martin, Assistant Professor of Computer Science

The first part of the project was to develop a graphics package to demonstrate pressure distribution on airfoils. One dimensional color contours and two-dimensional mesh surfaces are available. Major approaches are McAllister's algorithm for shape-preserving quadratic splines, Coon's bivariate interpolating scheme and Jensen's removal of hidden line algorithm.

The second part of the project was to design a graphics system for visualization of a solid obstacle immersed in a set of convective translucent density particles. Volumes of density particles are renormalized to apply geometric optics. The ray tracing mechanism is replaced by a model of sequences of renormalized planes to avoid the high cost of solving complicated differential equations. The shape of the obstacle surface is recovered by constructing individual microsurfaces from input density data. A Gaussian normal distribution is applied to approximate the light reflection rate on the obstacle surface.

Task Decomposition for Multiple Robot Arms

Jerrold L. Marco, Masters Candidate in Computer Science
Worthy N. Martin, Assistant Professor of Computer Science

Industrial robots are currently being used for a wide variety of tasks. Almost always, however, an individual robot is programmed to perform a single particular task. It makes sense to explore the concept of parallel task execution by multiple robots. This presupposes a method for decomposing tasks.

We would like to be able to give a computer which controls a number of robots a high level description of a task, and allow it to decompose the task for the robots currently at its disposal, and to construct the actual low level robot programs.

A number of issues are raised by this research. Is it more complicated to decompose tasks for larger numbers of robots, or is there a general solution? What relationship does this problem bear to the problem of automatically finding parallelism within programs? Can we find a method which is time and space efficient enough to operate "on the fly," or will we have to decompose any task of interest to us in advance, before any physical robots actually come into play?

The goal of the project is to construct a working system which will automate the decomposition of tasks. The system will then be evaluated as to its effectiveness. A part of the evaluation will concern what constraints it was found necessary to place upon the problem in order to be able to construct such a system at all. An open issue is just what criteria will be used to evaluate the system. At the very least, we would like to have an existence proof that such a system can indeed be built.

Vectorized Incomplete Conjugate Gradient

Eugene L. Poole, Ph.D. Candidate in Applied Mathematics
James Ortega, Professor of Applied Mathematics

The incomplete Cholesky conjugate gradient method is an iterative method for solving large, sparse linear systems of equations. We assume that the coefficient matrix is symmetric and positive definite. Though the incomplete Cholesky preconditioning has been used quite successfully on serial computers, in its usual form the factorization and forward and back solves necessary to implement the method do not vectorize with suitably long vectors on machines like the Cyber 205. We have demonstrated that with multi-coloring orderings and diagonal storage this limitation can be overcome.

During the past two years we have implemented incomplete Cholesky conjugate gradient on the Cyber 205 for three different two dimensional model problems. Multi-color orderings and diagonal storage of the matrix give vector lengths that are proportional to the number of unknowns in both the decomposition and forward and back solves necessary in the implementation of incomplete Cholesky preconditioning. The purpose of the current research is to extend and generalize the results already obtained in two dimensional problems to three dimensional structural problems.

We are currently considering a simplified model of a space platform. A program is running on the Cyber 205 at Langley Research Center which applies the multi-coloring strategy and diagonal storage to this problem. Our goal is to develop algorithms and strategies so that multi-coloring can be easily applied to the types of problems of interest in large scale scientific computing. It is hoped that theoretical results will be obtained to serve as guidelines in applying the multi-coloring techniques and the incomplete Cholesky preconditioning to a variety of supercomputing applications.

Iterative Methods for Solving Linear Equations on the Flex/32

Courtenay T. Vaughan, Masters Candidate in Computer Science
James M. Ortega, Professor of Applied Mathematics

In order to speedup the rate of computation of computers, parallel computers which have several processors working concurrently, are being developed. In order to take advantage of these computers, however, numerical methods have to be modified to run efficiently in parallel.

This project will first examine the implementation of several iterative methods for solving linear equations on the Flex/32, a parallel computer with up to 20 processors. The iterative methods which will be studied at first are Jacobi's method and SOR, using Poisson's problem on a unit square as a model problem. These methods will be implemented both in a synchronous manner in which all of the processors work on the same iteration at the same time and in an asynchronous manner which should be faster since it eliminates the overhead of synchronization. After these methods have been compared for the first model problem, the plane stress problem will be considered.

The second area of research will be to implement the conjugate gradient method and preconditioned conjugate gradient in parallel on the Flex/32. These methods will first be implemented in a synchronous manner and then in an asynchronous manner since it is not known if asynchronous conjugate gradient will converge. The preconditioner used for preconditioned conjugate gradient will be m steps of SSOR.

Finally, these methods will be implemented as part of a finite element code for the Flex/32.